PYTHON

Master Data Science & IA

Institut Supérieur Informatique

Mail: sybrahima31@gmail.om
Link : https://github.com/syibrahima31

Sy ibrahima

Manipulation de fichiers

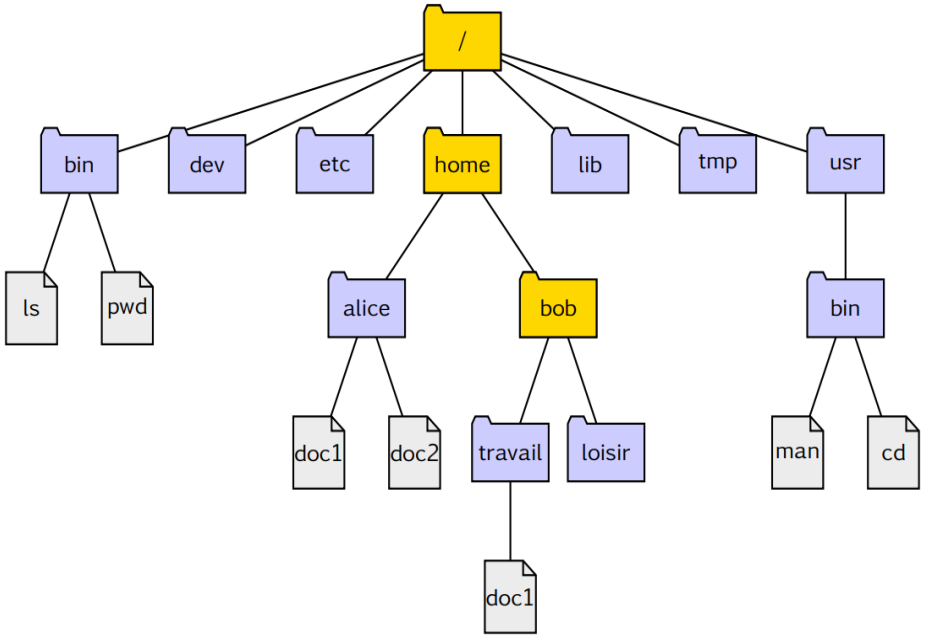
Le module *os*

Les instructions permettant à l'interprète de dialoguer avec le système d'exploitation font partie du module **os** :

```
>>> import os
```

La fonction **getcwd()** indique le répertoire courant :

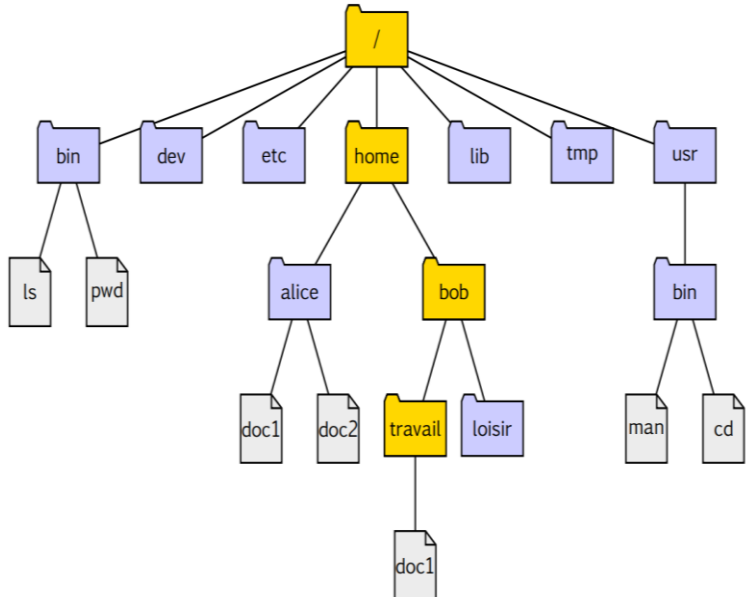
```
>>> os.getcwd()
'/home/bob'
```



Le module *os*

La fonction **chdir** permet de changer le répertoire courant :

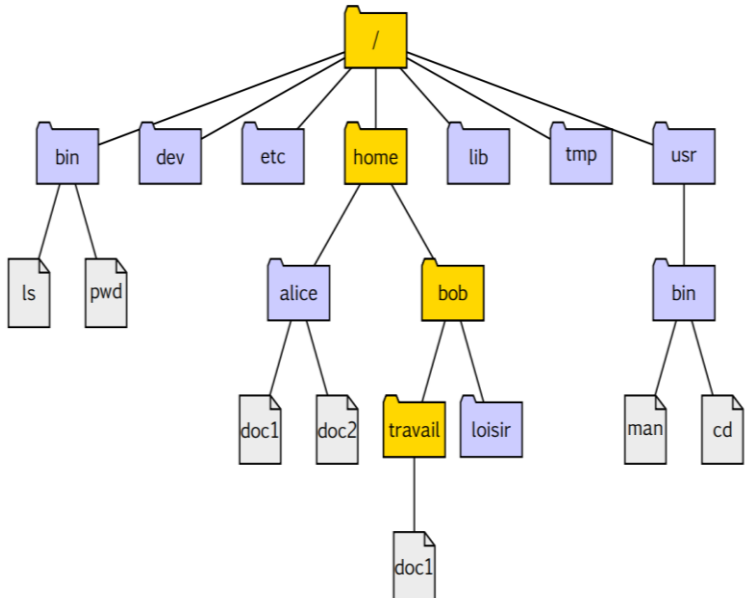
```
>>> os.chdir('/home/bob/travail')
```



Le module *os*

La fonction **listdir** liste le contenu d'un répertoire :

```
>>> os.listdir('/home/bob/travail')
['doc1']
```



Lecture d'un fichier texte

La fonction **open** propose trois modes d'ouverture d'un fichier :

- en lecture ('r')
- en écriture ('w')
- en ajout ('a')

Pour ouvrir en ***lecture*** le fichier ***exemple.txt*** du répertoire courant :

```
>>> comptine = open('exemple.txt', 'r')
```

Nous venons de créer un objet **comptine** faisant référence au fichier **exemple.txt** :

```
>>> comptine
<_io.TextIOWrapper name='exemple.txt' mode='r' encoding='UTF-8'>
```

Cet objet est un flux : les caractères sont lisibles uniquement les uns après les autres, sans possibilité de retour en arrière ni de saut en avant.

Lecture d'un fichier texte

Pour lire le fichier dans son entier : la méthode **read()**.

```
>>> comptine.read()
'Am, stram, gram,\nPic et pic et colégram,\nBour et bour et
ratatam,\nAm, stram, gram.'
```

Pour lire le fichier par groupe de 10 caractères :

```
>>> lst = []
>>> while True:
... txt = comptine.read(10)
... if len(txt) == 0:
... break
... lst.append(txt)
>>> lst
['Am, stram,', ' gram,\nPic', ' et pic et', ' colégram,', '\nBour
et b', 'our et rat', 'atam,\nAm, ', 'stram, gra', 'm.']
```

Lecture d'un fichier texte

Pour lire le fichier ligne par ligne : la méthode **readlines(n)**

```
>>> comptine.readlines()
['Am, stram, gram,\n', 'Pic et pic et colégram,\n',
'Bour et bour et ratatam,\n', 'Am, stram, gram.']
```

Lecture par énumération des lignes :

```
>>> n = 0
>>> for l in comptine:
... n += 1
... print('{} :'.format(n), l, end='')
1 : Am, stram, gram
2 : Pic et pic et colégram,
3 : Bour et bour et ratatam,
4 : Am, stram, gram.
```

Pour fermer un fichier : la méthode **close().**

```
>>> comptine.close()
```

Fichiers CSV

On considère le fichier **planetes.txt** contenant le texte suivant :

```
Mercure, 2439, 3.7, 88
Vénus, 6052, 8.9, 225
Terre, 6378, 9.8, 365
Mars, 3396, 3.7, 687
```

Chaque ligne est découpée en colonnes par la méthode **split** :

```
>>> tab = []
>>> for chn in lignes:
... tab.append(chn.split(','))
```

À cette étape, tab est une liste de listes égale à :

```
[['Mercure', ' 2439', ' 3.7', ' 88\n'], ['Vénus', ' 6052', ' 8.9', ' 225\n'],
['Terre', ' 6378', ' 9.8', ' 365\n'], ['Mars', ' 3396', ' 3.7', ' 687\n']]
```

Fichiers CSV

On considère le fichier **planetes.txt** contenant le texte suivant :

```
Mercure, 2439, 3.7, 88
Vénus, 6052, 8.9, 225
Terre, 6378, 9.8, 365
Mars, 3396, 3.7, 687
```

Chaque ligne est découpée en colonnes par la méthode **split** :

```
>>> tab = []
>>> for chn in lignes:
... tab.append(chn.split(','))
```

À cette étape, tab est une liste de listes égale à :

```
[['Mercure', ' 2439', ' 3.7', ' 88\n'], ['Vénus', ' 6052', ' 8.9', ' 225\n'],
['Terre', ' 6378', ' 9.8', ' 365\n'], ['Mars', ' 3396', ' 3.7', ' 687\n']]
```

Fichiers CSV

On convertit les données numériques :

```
>>> for lst in tab:
...     lst[1] = int(lst[1])
...     lst[2] = float(lst[2])
...     lst[3] = int(lst[3])
```

La liste tab est maintenant prête à être utilisée :

```
[['Mercure', 2439, 3.7, 88], ['Vénus', 6052, 8.9, 225], ['Terre', 6378, 9.8,
365], ['Mars', 3396, 3.7, 687]]
```

Écrire dans un fichier texte

- Deux modes d'ouverture possibles : le mode 'w' **(write)** et le mode 'a' **(append).** Dans les deux cas, la méthode **write** permet d'enregistrer les chaînes de caractères passées en argument les unes à la suite des autres.

- Le mode 'a' pour ajouter au fichier **planetes.txt** des données supplémentaires :

```
>>> planetes = open('planetes.txt', 'a')
>>> planetes.write('Jupiter, 71492, 24.8, 4335\n')
>>> planetes.write('Saturne, 60268, 10.4, 10757\n')
>>> planetes.close()
```

Le fichier **planetes.txt** contient maintenant le texte suivant :

```
Mercure, 2439, 3.7, 88
Vénus, 6052, 8.9, 225
Terre, 6378, 9.8, 365
Mars, 3396, 3.7, 687
Jupiter, 71492, 24.8, 4335
Saturne, 60268, 10.4, 10757
```

Encodage d'un fichier texte

Le jeu de caractères ascii

Historiquement un caractère est codé sur **7 bits**, ce qui donne 2^7 = 128 caractères différents, qui constituent le jeu de caractères ascii.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0																
1																
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

L'espace a pour code ascii $(20)_{16}$ = 32 ; le caractère **'A'** a pour code ascii $(41)_{16}$= 65 ; le caractère **'a'** a pour code ascii (61)16 = 97.

Encodage d’un fichier texte

Le jeu de caractères ascii

Ce codage simple est insuffisant pour pouvoir représenter la diversité des caractères des langues autres que l’anglais, aussi un huitième bit a été utilisé pour ajouter au jeu de caractères ascii 128 autres caractères codés entre 128 = (80)16 et 255 = (ff)16.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
8																
9																
a		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯
b	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
c	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
d	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
e	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
f	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Encodage d’un fichier texte

Le jeu de caractères ascii

Chaque langue ayant des besoins spécifiques, ces extensions sont nombreuses et non compatibles entre elles : la norme latin-1 permet d’encoder les langues d’Europe occidentale, la norme latin-2 pour les langues d’Europe centrale, etc. Pas moins de 16 variantes existent pour le seul standard ISO 8859. Mais les écritures idéographiques comme le chinois nécessitent plusieurs milliers de caractères et ne peuvent donc être codées sur un seul octet.

La norme Unicode

Elle attribue un identifiant numérique universel à chacun des milliers de caractères nécessaires à la transcription des différentes langues. L'encodage le plus fréquent de l'Unicode est la norme utf-8 ; c'est la norme utilisée par défaut par Python.

Le fichier comptine.txt ouvert au format utf-8 :

```
>>> comptine = open('exemple.txt', 'r')
>>> print(comptine.read())
Am, stram, gram
Pic et pic et colégram,
Bour et bour et ratatam,
Am, stram, gram.
```

Le fichier comptine.txt ouvert au format latin-1 :

```
>>> comptine = open('exemple.txt', 'r', encoding='latin1')
>>> print(comptine.read())
Am, stram, gram
Pic et pic et colÃ©gram,
Bour et bour et ratatam,
Am, stram, gram.
```

L'encodage choisi n'est manifestement pas le bon !

La norme Unicode

La fonction **chr** retourne le caractère ASCII dont l'identifiant a été passé en paramètre mais aussi le caractère unicode associé à son identifiant :

```
>>> for i in range(945, 970):
... print(chr(i), end=' ')
α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω
```